Deep Reinforcement Learning

2025-56 Second

2022-23 Second Semester, M.Tech (AIML)

Session #2-3: Multi-armed Bandits

Instructors - DRL Course

Agenda for the class

- Recap
- k-armed Bandit Problem & its significance
- Action-Value Methods
  - Sample Average Method & Incremental Implementation
- Non-stationary Problem
- Initial Values & Action Selection

Acknowledgements: Some of the slides were adopted *with permission* from the course CSCE-689 (Texas A&M University) by Prof. Guni Sharon

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action
- **Objective** : to *maximize the expected total reward over some time*

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action
- **Objective** : to *maximize the expected total reward over some time*

Strategy:

- Identify the best lever(s)
- Keep pulling the identified ones

Questions:

- How do we define the *best ones*?
- What are the best levers?

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action
- **Objective** : to *maximize the expected total reward over some time*

$\mathbb{E}[a] = -\$0.5$

$\mathbb{E}[b] = -\$0.2$

$\mathbb{E}[c] = \$0.1$

$\mathbb{E}[d] = \$0.11$

- ***Expected Mean Reward*** for each action selected → call it ***Value*** of the action

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

K-armed Bandit Problem

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

- A_t - action selected on time step t
- Q_t (a) - estimated value of action a at time step t
- q_*(a) - value of an arbitrary action a

Note: If you knew the value of each action, then it would be trivial to solve the k -armed bandit problem: you would always select the action with highest value :-)

K-armed Bandit Problem

-1, -1, 5

$\hat{E}[a] = 1$

-0.2, -0.2

$\hat{E}[b] = -0.2$

-0.5, -0.5, -0.5

$\hat{E}[c] = -0.5$

-2, -2

$\hat{E}[d] = -2$

Keep pulling the levers; update the estimate of action values;

K-armed Bandit Problem

1. How to maintain the estimate of expected rewards for each action?

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

$$= \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

2. How to use the estimate $A_t \doteq \arg\max_a Q_t(a)$ action?

Greedy Action Selecti

K-armed Bandit Problem

2. How to use the estimate in selecting the right action?

   Greedy Action Selection

$$A_t \doteq \arg\max_a Q_t(a)$$

Actions which are inferior by the value estimate upto time t, could be indeed better than the greedy action at t !!!

3. Exploration vs. Exploitation?

ε-Greedy Action Selection / near-greedy action selection

Behave greedily most of the time; Once in a while, with small probability **ε** select randomly from among all the actions with equal probability, independently of the action-value estimates.

K-armed Bandit Problem

Greedy Action

{-1, -1, 5}

$N_{11}(a)=3$

$q_*(a)=-\$0.5$

$Q_{11}(a)=1$

{-0.2, -0.2}

$N_{11}(b)=2$

$q_*(b)=-\$0.2$

$Q_{11}(b)=-0.2$

{-0.5, -0.5, -0.5}

$N_{11}(c)=3$

$q_*(c)=\$0.1$

$Q_{11}(c)=-0.5$

{-2, -2}

$N_{11}(d)=2$

$q_*(d)=\$1$

$Q_{11}(d)=-2$

K-armed Bandit Problem

Action to Explore

{-1, -1, 5}

$N_{11}(a)=3$

$q_*(a)=-\$0.5$

$Q_{11}(a)=1$

{-0.2, -0.2}

$N_{11}(b)=2$

$q_*(b)=-\$0.2$

$Q_{11}(b)=-0.2$

{-0.5, -0.5, -0.5}

$N_{11}(c)=3$

$q_*(c)=\$0.1$

$Q_{11}(c)=-0.5$

{-2, -2}

$N_{11}(d)=2$

$q_*(d)=\$1$

$Q_{11}(d)=-2$

K-armed Bandit Problem

ε-Greedy Action Selection / near-greedy action selection

```
epsilon = 0.05 // small value to control exploration
def get_action():
    if random.random() > epsilon:
        return argmaxa(Q(a))
    else:
        return random.choice(A)
```

- **In the limit as the number of steps increases, every action will be sampled by ε-greedy action selection an infinite number of times. This ensures that all the Qt (a) converge to q_* (a).**
- **Easy to implement / optimize for epsilon / yields good results**

Ex-1: In ε-greedy action selection, for the case of two actions and ε = 0.5, what is the probability that *the greedy action* is selected?

p (greedy action)

= p (greedy action AND greedy selection) + p (greedy action AND random selection)

= p (greedy action | greedy selection) p (greedy selection)
+ p (greedy action | random selection) p (random selection)

= p (greedy action | greedy selection) (1-**ε**) + p (greedy action | random selection) (**ε**)

= p (greedy action | greedy selection) (0.5) + p (greedy action | random selection) (0.5)

= (1) (0.5) + (0.5) (0.5)

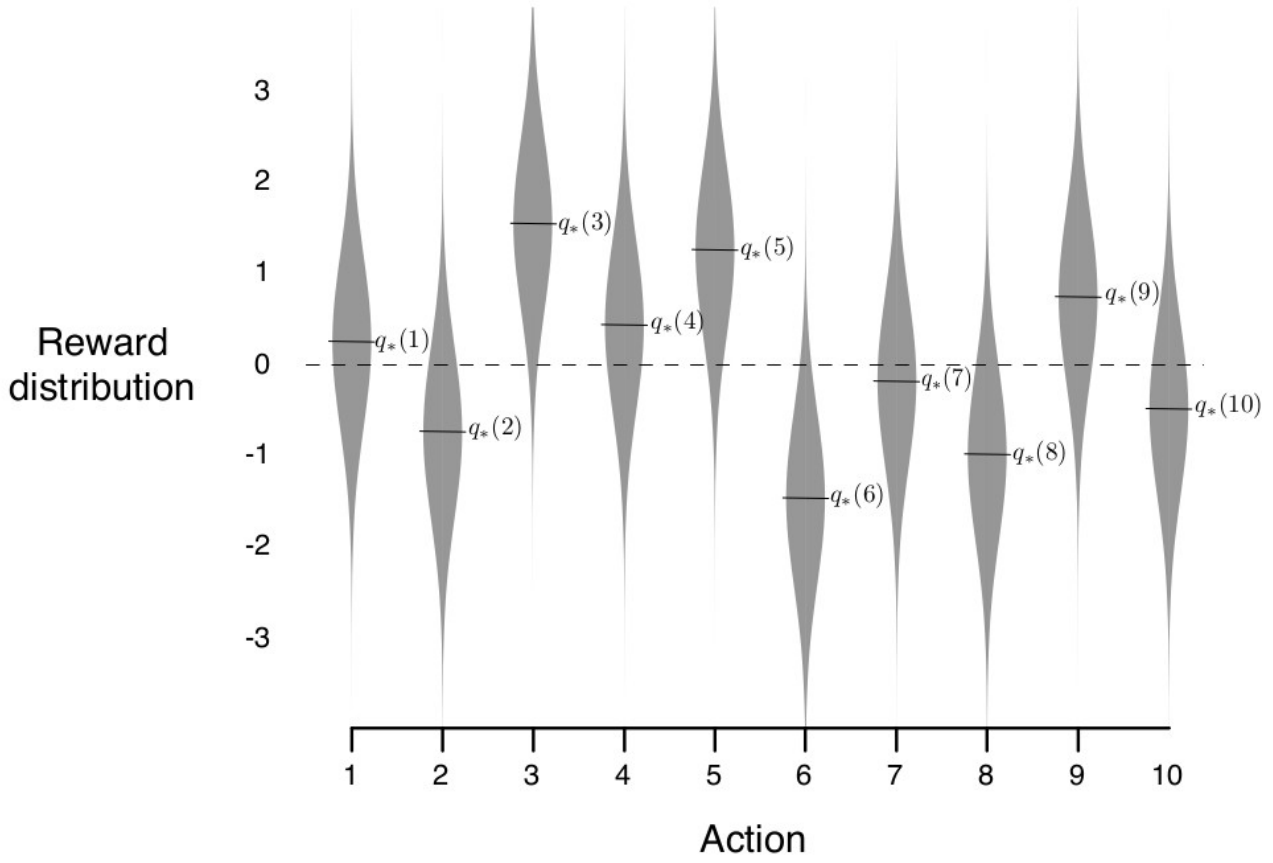
= 0.5 + 0.25

= 0.75

10-armed Testbed

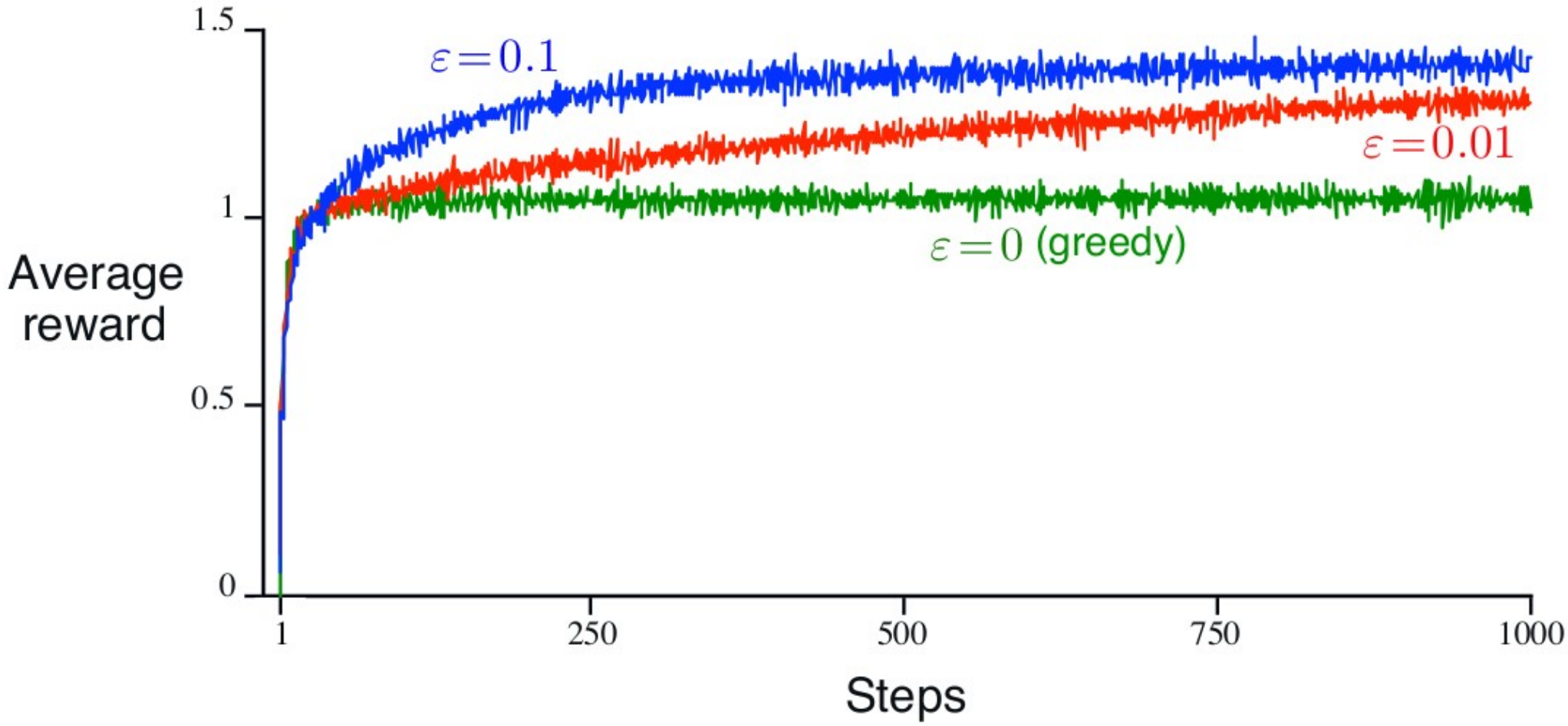
Example:

- A set of 2000 randomly generated k -armed bandit problems with k = 10
- Action values were selected according to a normal (Gaussian) distribution with mean 0 and variance 1.
- While selecting action A_t at time step t, the actual reward, R_t , was selected from a normal distribution with mean $q_*(A_t)$ and variance 1
- One Run : Apply a method for 1000 time steps to one of the bandit problems
- Perform 2000 runs, each run with a different bandit problem, to get an algorithms average behavior

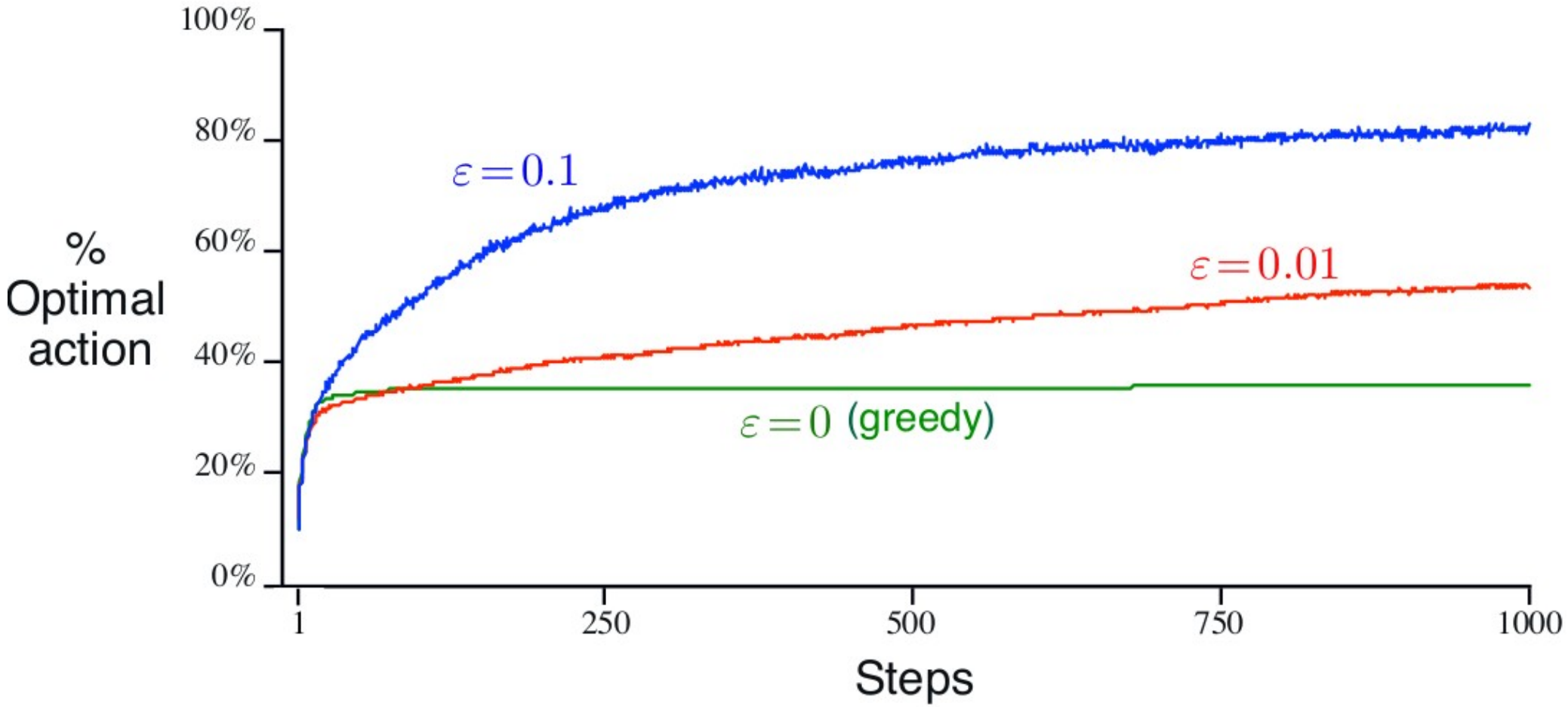


An example bandit problem from the 10-armed testbed

Average performance of ε-greedy action-value methods on the 10-armed testbed



Average performance of ε-greedy action-value methods on the 10-armed testbed



Ex-2:

Consider a k -armed bandit problem with k = 4 actions, denoted 1, 2, 3, and 4.

Consider applying to this problem a bandit algorithm using **ε**-greedy action selection, sample-average action-value estimates, and initial estimates of Q1 (a) = 0, for all a.

Suppose the initial sequence of actions and rewards is A1 = 1, R1 = 1, A2 = 2, R2 = 1, A3 = 2, R3 = 2, A4 = 2, R4 = 2, A5 = 3, R5 = 0.

On some of these time steps the **ε** case may have occurred, causing an action to be selected at random.

On which time steps did this definitely occur? On which time steps could this possibly have occurred?

Solution is given below:

Initially,

$$Q_1(1) = Q_1(2) = Q_1(3) = Q_1(4) = 0$$

Time	Q-values before selection	Greedy action(s)	Selected action	Conclusion
$t = 1$	All values are 0	1, 2, 3, 4	1	Could be greedy or random
$t = 2$	$Q(1) = 1$, others 0	1	2	**Definitely random**
$t = 3$	$Q(1) = 1,\ Q(2) = 1$	1 or 2	2	Could be greedy or random
$t = 4$	$Q(2) = 1.5,\ Q(1) = 1$	2	2	Could be greedy or random
$t = 5$	$Q(2) = \frac{1+2+2}{3} = \frac{5}{3}$	2	3	**Definitely random**

Incremental Implementation

- Efficient approach to compute the estimate of action-value;

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

- Given Qn and the nth reward, Rn , the new average of all n rewards can be computed as follows

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}$$

Incremental Implementation

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}
$$

Note:

- StepSize decreases with each update
- We use $\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_t(a)$ to denote step size (constant / varies with each step)

Discussion:

Const vs. Variable step size?

$$NewEstimate \leftarrow OldEstimate + StepSize\Big[Target - OldEstimate\Big]$$

Discussion on Exploration vs. Exploitation

1. What if the reward variance is
   A)larger, say 10 instead of 1?

   B) zero ? [deterministic]
2. What if the bandit task is non-stationary? [that is, the true values of the actions changed over time]

Solution: a) First, suppose the **reward variance becomes large**, for example 10 instead of 1. This means the reward from the same action can fluctuate a lot. So, even a good action may sometimes give a poor reward and a bad action may occasionally give a high reward. Therefore, the agent needs **more exploration and more samples** before it can confidently identify the best action.

b) Now suppose the **reward variance is zero**. In this case, the environment is deterministic. Every time we select an action, it gives exactly the same reward. So, after trying each action once, we can almost immediately know which action is best. Very little further exploration is required, and the agent can mainly exploit the best action.

c) consider a **non-stationary bandit**, where the true action values change with time. Here, old observations may no longer represent the current situation. Therefore, the agent must **continue exploring throughout learning** rather than permanently selecting one action. Methods such as **constant step-size updates** are useful because they give more importance to recent rewards.

- Use a **constant step-size** α: $Q_{n+1} = Q_n + \alpha(R_n - Q_n)$, which gives exponentially decaying weight to older rewards, tracking the changing values.
- Keep **exploring permanently** (don't let $\varepsilon \to 0$), since a previously poor arm may become the best one

- So, in short: **high variance requires more exploration, zero variance requires less exploration, and a changing environment requires continuous exploration.**”

Bandit Algorithm with Incremental Update/ ε-greedy selection

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Given: 3 actions A_1, A_2, A_3, $\epsilon = 0.2$

- **80% probability → Exploit:** choose action with highest $Q(a)$
- **20% probability → Explore:** choose a random action
- Initially: $Q(A_1) = Q(A_2) = Q(A_3) = 0$

Incremental Update

$$\boxed{Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]}$$

Step	Selection	Action	Reward R	Calculation	New $Q(A)$
1	Initial/Random	A_1	6	$0 + \frac{1}{1}(6 - 0)$	6.00
2	Exploit	A_1	8	$6 + \frac{1}{2}(8 - 6)$	7.00
3	**Explore**	A_2	10	$0 + \frac{1}{1}(10 - 0)$	10.00
4	Exploit	A_2	6	$10 + \frac{1}{2}(6 - 10)$	8.00
5	Exploit	A_2	9	$8 + \frac{1}{3}(9 - 8)$	8.33

Final Estimates

$$Q(A_1) = 7, \qquad \boxed{Q(A_2) = 8.33}, \qquad Q(A_3) = 0$$

Conclusion: A_2 is currently the **best action.** Exploration helped discover A_2, even though A_1 initially appeared best.

Key point: Incremental update calculates the new average reward **without storing all previous rewards.**

What are problems whose solutions are modelled as MAB?

Non-stationary Problem

- Most RL problems are non-stationary !
- Give more weight to recent rewards than to long-past rewards !!!

$$Q_{n+1} \doteq Q_n + \alpha\Big[R_n - Q_n\Big]$$

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2\alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i .
\end{aligned}$$

Exponential recency-weighted average

Optimistic Initial Values

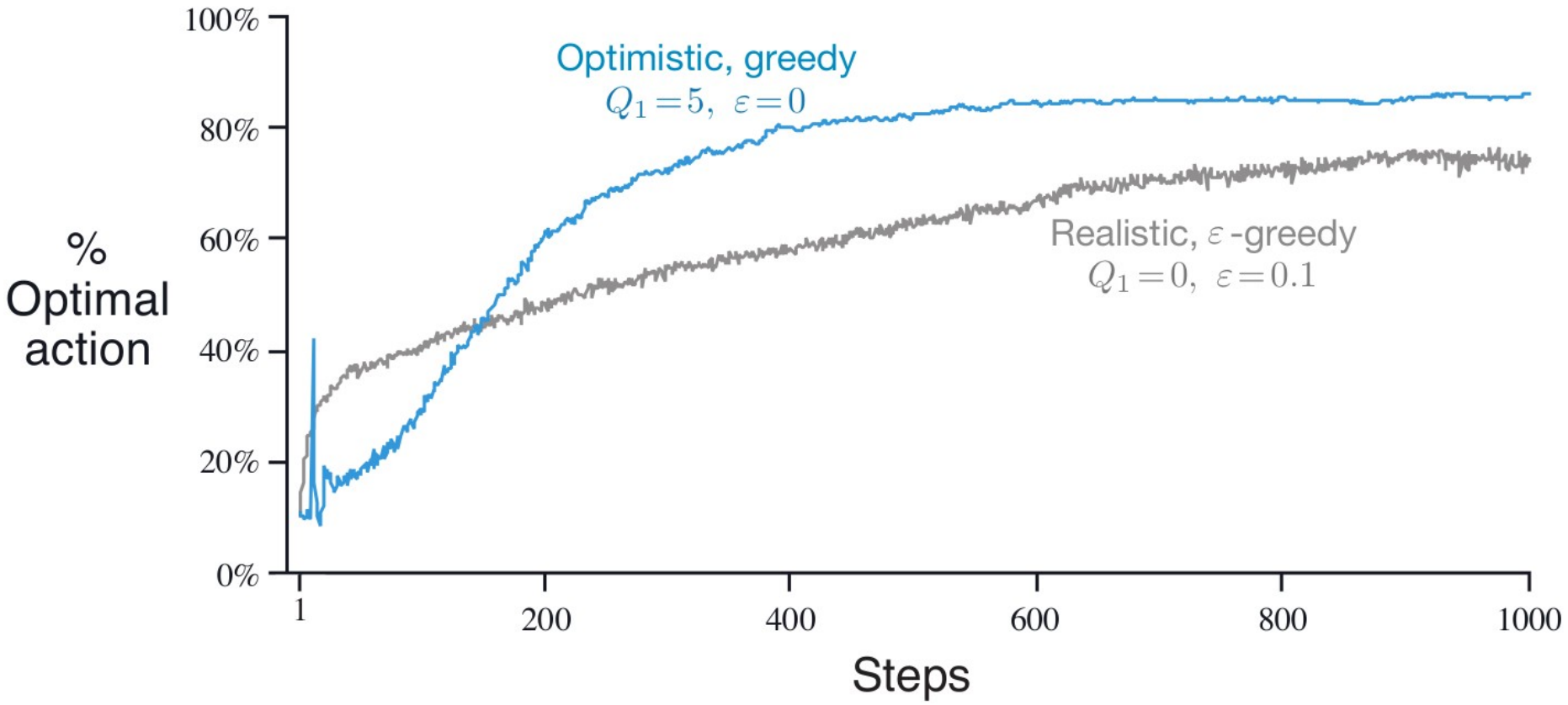
- All the above discussed methods are ***biased*** by their initial estimates
- For sample average method the bias disappears once all actions have been selected at least once
- For methods with constant α, the bias is permanent, though decreasing over time
- Initial action values can also be used as a simple way of encouraging exploration.
- In 10 armed testbed, set initial estimate to +5 rather than 0.

  This can encourage action-value methods to explore.

  Whichever actions are initially selected, the reward is less than the starting estimates;

  the learner switches to other actions, being disappointed with the rewards

Optimistic Initial Values



Caution:
Optimistic Initial Values can only be considered as a simple trick that can be quite effective on stationary problems, but it is far from being a generally useful approach to encouraging exploration.

Question:
Explain how in the non-stationary scenario the optimistic initial values will fail (to explore adequately).

The effect of optimistic initial action-value estimates on the 10-armed testbed.
Both methods used a constant step-size parameter, $\alpha = 0.1$

Upper-Confidence-Bound Action Selection

- Each time a is selected the uncertainty is presumably reduced
- Each time an action other than a is selected, t increases but $N_t(a)$ does not; because t appears in the numerator, the uncertainty estimate increases.
- Actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time

Action Value at time t for a

Confidence Level

Measure of Uncertainty

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

UCB Formula:

$$UCB(a) = Q(a) + c\sqrt{\frac{\ln t}{N(a)}}$$

Assume:

- Total trials: $t = 10$
- Confidence parameter: $c = 2$

Action	Average Reward $Q(a)$	Times Selected $N(a)$	Exploration Bonus	UCB Value
A	8	5	$2\sqrt{\frac{\ln 10}{5}} = 1.36$	**9.36**
B	6	3	$2\sqrt{\frac{\ln 10}{3}} = 1.75$	**7.75**
C	5	2	$2\sqrt{\frac{\ln 10}{2}} = 2.15$	**7.15**

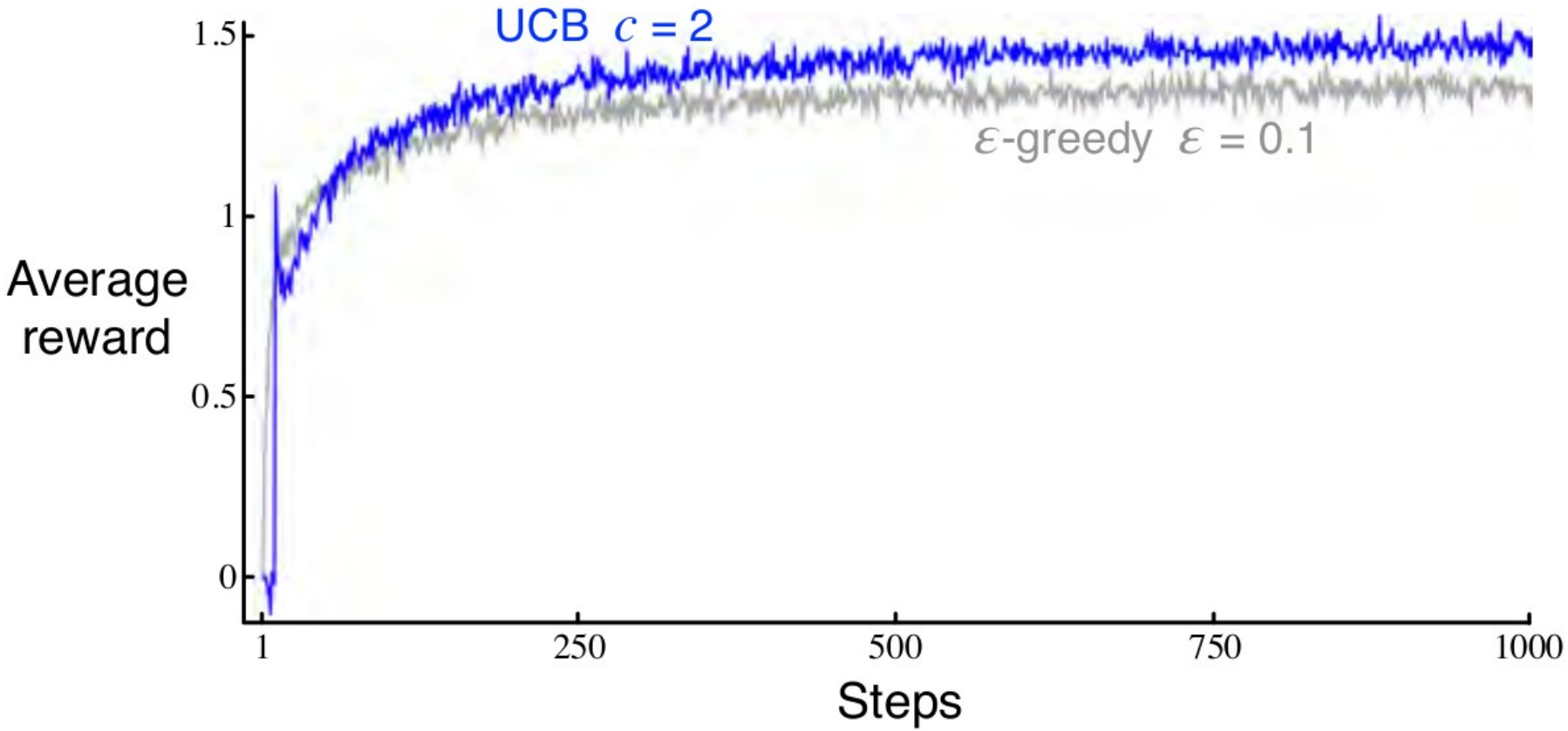
Decision

Select Action A because it has the highest UCB value (9.36).

Key Observation

- **Action A:** High reward, low uncertainty → Small exploration bonus.
- **Action C:** Low reward, high uncertainty → Large exploration bonus.
- **UCB balances exploitation (high reward) and exploration (high uncertainty) automatically.**

Upper-Confidence-Bound Action Selection



UCB often performs well, as shown here, but is more difficult than "-greedy to extend beyond bandits to the more general reinforcement learning settings

Policy-based algorithms

- Forget about action-value (Q) estimates, we don't really care about them
- We care about what actions to chose

Let's assign a preference to each action and tweak its value

- Define $H_t(a)$ as a numerical preference value associated with action a
- Which action is selected?
  - $A_t = \underset{a}{\text{argmax}}[H_t(a)]$
    - Hardmax results in no exploration -- deterministic action selection

Softmax!

Softmax function

- **Input**: vector of preferences
- **Output**: vector of probabilities forming a valid distribution
- $\Pr\{a_t = a\} = \frac{e^{H_t(a)}}{\sum_{a' \in A} e^{H_t(a')}} = \Pr(a)$
- $H_t \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{bmatrix} 6 \\ 9 \\ 2 \end{bmatrix}$
- $\text{softmax} \begin{pmatrix} 6 \\ 9 \\ 2 \end{pmatrix} = \begin{bmatrix} 0.047 \\ 0.952 \\ 0.00087 \end{bmatrix}$
- That is, with $\Pr(0.95)$ choose a_2, $\Pr(0.05)$ choose a_1, and $< \Pr(0.01)$ choose a_3

Softmax function

- Exploration – checked!
- Softmax provides another important attribute – **a differentiable policy**
- Say that we learn that action a_1 results in good relative performance
- Hardmax: $A_t = \underset{a}{\text{argmax}}[H_t(a_1), H_t(a_2), H_t(a_3)]$
  - Change $H(a_1)$ such that $\Pr(a_1)$ is increased, $\frac{\partial \Pr(a_1)}{\partial H(a_1)} = NA$
- Softmax: $\Pr(a_1) = \frac{e^{H_t(a)}}{\sum_{a' \in A} e^{H_t(a')}}$
  - Change $H(a_1)$ such that $\Pr(a_1)$ is increased -> update towards $\frac{\partial \Pr(a_1)}{\partial H(a_1)}$

Gradient ascend

- $H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})\frac{\partial \Pr(A_t)}{\partial H(A_t)}$?
- $\forall a \neq A_t, H_{t+1}(a) = H_t(a) + \alpha(R_t - \overline{R_t})\frac{\partial \Pr(A_t)}{\partial H(a)}$
- Update the preferences based on observed reward and a baseline reward ($\overline{R_t}$)
- If the observed reward is larger than the baseline:
  - Increase the preference of the chosen action, A_t
  - Decrease the preference of all other actions, $\forall a \neq A_t$
- Else do the opposite

1. Start with Softmax probability

For the selected action A_t:

$$Pr(A_t) = \frac{e^{H_t(A_t)}}{\sum_b e^{H_t(b)}}$$

2. Take log

$$\log Pr(A_t) = \log\left(\frac{e^{H_t(A_t)}}{\sum_b e^{H_t(b)}}\right)$$

Using $\log(x/y) = \log x - \log y$:

$$\boxed{\log Pr(A_t) = H_t(A_t) - \log\left(\sum_b e^{H_t(b)}\right)}$$

3. Derivative for the selected action A_t

Differentiate with respect to $H_t(A_t)$:

$$\frac{\partial \log Pr(A_t)}{\partial H_t(A_t)} = 1 - \frac{e^{H_t(A_t)}}{\sum_b e^{H_t(b)}}$$

Since

$$\frac{e^{H_t(A_t)}}{\sum_b e^{H_t(b)}} = Pr(A_t)$$

we get:

$$\boxed{\frac{\partial \log Pr(A_t)}{\partial H_t(A_t)} = 1 - Pr(A_t)}$$

$$\boxed{H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)[1 - Pr(A_t)]}$$

4. Derivative for any other action $a \neq A_t$

Differentiate the same log probability with respect to $H_t(a)$:

$$\frac{\partial \log Pr(A_t)}{\partial H_t(a)} = 0 - \frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}}$$

Since

$$\frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}} = Pr(a)$$

we get:

$$\boxed{\frac{\partial \log Pr(A_t)}{\partial H_t(a)} = -Pr(a)}$$

Therefore,

$$\boxed{H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)Pr(a), \qquad a \neq A_t}$$

Update Rule

On each step, after selecting action A t and receiving the reward Rt , Update the action preferences :

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})(1 - \Pr(A_t))$$
$$\forall a \neq A_t, H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R_t}) \Pr(a)$$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

Numerical Example: Gradient Bandit Update — 2 Iterations

Given: 3 actions A_1, A_2, A_3, learning rate $\alpha = 0.1$

Initially:

$$H(A_1) = H(A_2) = H(A_3) = 0, \qquad \pi(A_1) = \pi(A_2) = \pi(A_3) = \mathbf{0.333}$$

Update Rules

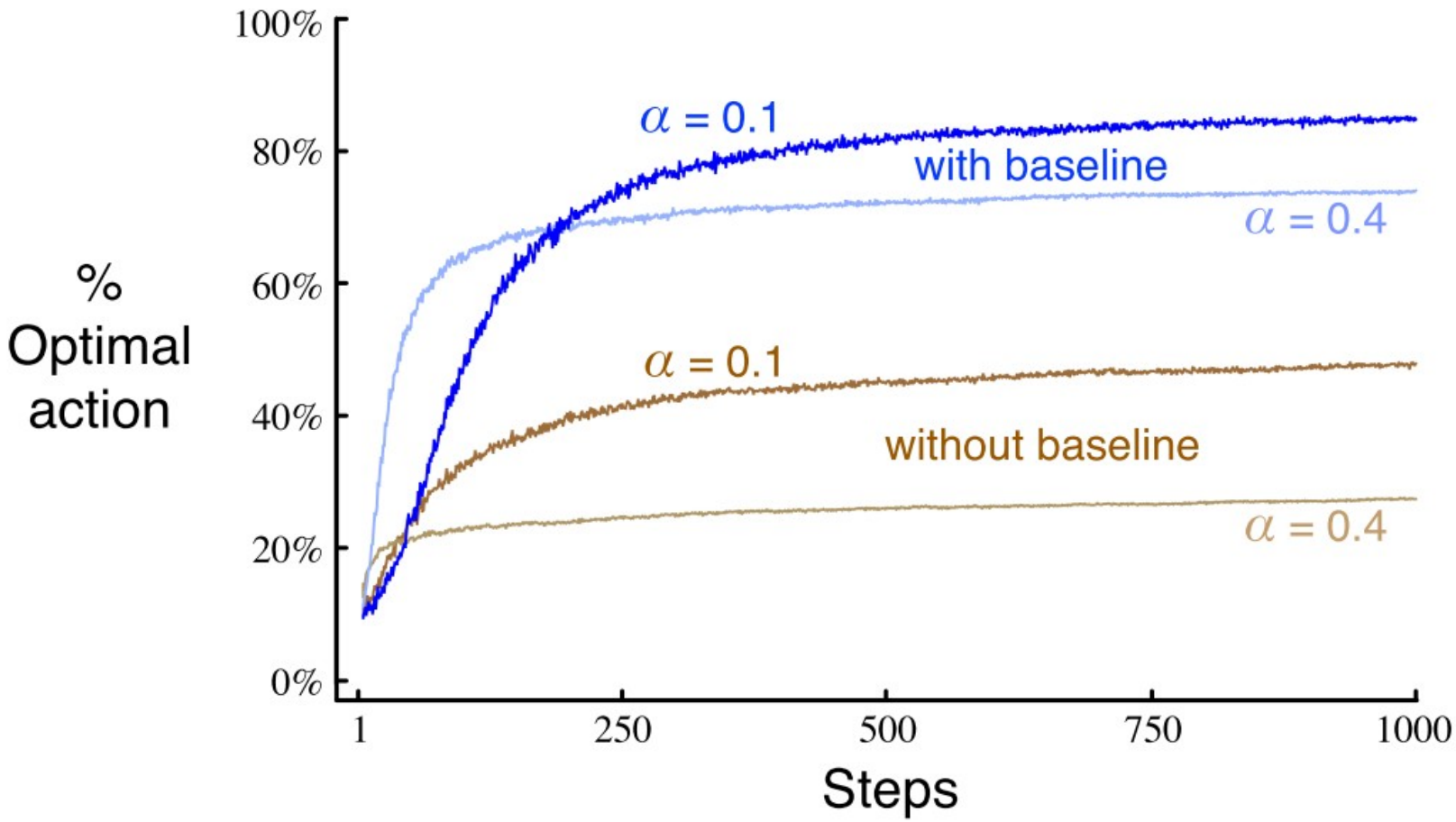
$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a), \quad a \neq A_t$$

	Iteration 1	Iteration 2
Selected action	A_1	A_1
Reward	$R_1 - 8$	$R_2 - 4$
Average reward	$\bar{R}_1 - 5$	$\bar{R}_2 - 6$
Difference	$8 - 5 = +3$	$4 - 6 = -2$
$H(A_1)$	$0 + 0.1(3)(1 - 0.333) - \mathbf{0.200}$	$0.2 + 0.1(-2)(1 - 0.403) - \mathbf{0.081}$
$H(A_2)$	$0 - 0.1(3)(0.333) - \mathbf{-0.100}$	$-0.1 - 0.1(-2)(0.299) - \mathbf{-0.040}$
$H(A_3)$	$0 - 0.1(3)(0.333) - \mathbf{-0.100}$	$-0.1 - 0.1(-2)(0.299) - \mathbf{-0.040}$

After **Iteration 1**, new Softmax probabilities are:

$$\boxed{\pi(A_1) = 0.403, \quad \pi(A_2) = 0.299, \quad \pi(A_3) = 0.299}$$

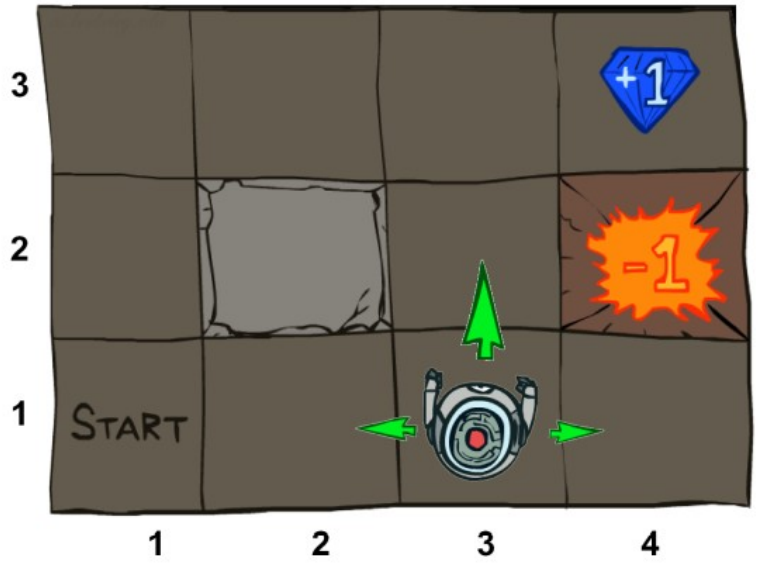

100%
80%
60%
40%
20%
0%
% Optimal action
α = 0.1
with baseline
α = 0.4
α = 0.1
without baseline
α = 0.4
1
250
500
750
1000
Steps

What did we learn?

- **Problem**: choose the action that results in highest expected reward
- **Assumptions**: 1. actions' expected reward is unknown, 2. we are confronted with the same problem over and over, 3. we are able to observe an action's outcome once chosen
- **Approach**: learn the actions' expected reward through exploration (value based) or learn a policy directly (policy based), exploit learnt knowledge to choose best action
- **Methods:** 1. greedy + initializing estimates optimistically, 2. epsilon-greedy, 3. Upper-Confidence-Bounds, 4. gradient ascend + soft-max

A different scenario

- Associative vs. Non-associative tasks ?
- Policy: A mapping from situations to the actions that are best in those situations
- (discuss) How do we extend the solution for non-associative task to an associative task?
  - **Approach**: Extend the solutions to non-stationary task to non-associative tasks
    - Works, if the true action values changes slowly
  - What if the context switching between the situations are made explicit?
    - How?
    - Need Special approaches !!!



Required Readings

1. Chapter-2 of Introduction to Reinforcement Learning,2nd Ed., Sutton & Barto
2. A Survey on Practical Applications of Multi-Armed and Contextual Bandits, Djallel Bouneffouf , Irina Rish [https://arxiv.org/pdf/1904.10040.pdf]

Thank you !